


# C# 프로그래밍

## - 9 주차 (1강)



- 가변 배열
- 컬렉션

- ArrayList, Queue, Stack, Hashtable

- 컬렉션 초기화

# 가변 배열 (Jagged Array)

- 다양한 길이의 배열을 요소로 갖는 다차원 배열
  - ✓ 가변 배열의 요소로 입력되는 배열은 그 길이가 모두 같을 필요가 없음

가변 배열 선언 형식:

```
데이터형식[ ][ ] 배열이름 = new 데이터형식[가변 배열의 용량][ ];
```

가변 배열 선언 예:

```
int[ ][ ] arr = new int[5][ ]; // 배열의 용량 5
arr[0] = new int[10]; // 0번 요소에는 길이가 10인 배열 생성
arr[1] = new int[9]; // 1번 요소에는 길이가 9인 배열 생성
arr[2] = new int[8]; // 2번 요소에는 길이가 8인 배열 생성
arr[3] = new int[3]; // 3번 요소에는 길이가 3인 배열 생성
arr[4] = new int[5]; // 4번 요소에는 길이가 5인 배열 생성
```

int [10]	arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]	arr[0][4]	arr[0][5]	arr[0][6]	arr[0][7]	arr[0][8]	arr[0][9]
int [9]	arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]	arr[1][4]	arr[1][5]	arr[1][6]	arr[1][7]	arr[1][8]	
int [8]	arr[2][0]	arr[2][1]	arr[2][2]	arr[2][3]	arr[2][4]	arr[2][5]	arr[2][6]	arr[2][7]		
int [3]	arr[3][0]	arr[3][1]	arr[3][2]							
int [5]	arr[4][0]	arr[4][1]	arr[4][2]	arr[4][3]	arr[4][4]					



# 가변 배열 초기화

- 다양한 길이의 배열을 요소로 갖는 다차원 배열

가변 배열 선언 형식:

```
데이터형식[][] 배열이름 = new 데이터형식[가변 배열의 용량][];
```

가변 배열 선언 후 각 요소에 1차원 배열 생성

```
int[][] jagged = new int[3][];           // 배열의 용량 3
jagged[0] = new int[5] { 1, 2, 3, 4, 5 }; // 0번 요소에는 길이가 5인 배열
jagged[1] = new int[] { 10, 20, 30 };     // 1번 요소에는 길이가 3인 배열
jagged[2] = new int[] { 100, 200 };       // 2번 요소에는 길이가 2인 배열
```

가변 배열 선언과 동시에 초기화 가능

```
int[][] jagged2 = new int[2][] {
    new int[] {1000,2000},
    new int[4] {6,7,8,9}
};
```



# 가변 배열 예제 코드

```
static void Main(string[] args)
{
    int[][] jagged = new int[3][]; // 배열의 용량 3
    jagged[0] = new int[5] { 1, 2, 3, 4, 5 }; // 0번 요소에는 길이가 5인 배열
    jagged[1] = new int[] { 10, 20, 30 }; // 1번 요소에는 길이가 3인 배열
    jagged[2] = new int[] { 100, 200 }; // 2번 요소에는 길이가 2인 배열
    PrintElement(jagged);

    Console.WriteLine();

    int[][] jagged2 = new int[2][] {
        new int[] { 1000, 2000 },
        new int[4] { 6, 7, 8, 9 } };
    PrintElement(jagged2);
}
```

출력 결과:

Length : 5, 1 2 3 4 5

Length : 3, 10 20 30

Length : 2, 100 200

Length : 2, 1000 2000

Length : 4, 6 7 8 9

```
void PrintElement(int[][] tmpJagged)
{
    foreach (int[] arr in tmpJagged)
    {
        Console.Write($"Length : {arr.Length}, ");
        foreach (int e in arr)
        {
            Console.Write($"{e} ");
        }
        Console.WriteLine();
    }
}
```





# 컬렉션 (Collection)

- 같은 성격을 띤 데이터의 모음을 담는 자료구조
  - ✓ 배열도 .NET 이 제공하는 다양한 컬렉션 자료구조의 일부
  - ✓ ICollection 인터페이스를 상속

```
public abstract class Array : ICloneable,  
    IList, ICollection, IEnumerable
```

- .NET 이 제공하는 주요 컬렉션 자료구조
  - ✓ ArrayList
  - ✓ Queue
  - ✓ Stack
  - ✓ Hashtable



# ArrayList

- 가장 배열과 닮은 컬렉션
  - ✓ 컬렉션의 요소에 접근할 때는 [ ] 연산자 이용
  - ✓ 특정 위치에 있는 요소에 데이터를 임의로 할당 가능
- 배열과 달리 필요에 따라 자동으로 용량이 늘어나거나 줄어듦
- ArrayList 의 주요 메소드
  - ✓ Add() : 가장 마지막에 있는 요소 뒤에 새 요소를 추가
  - ✓ RemoveAt() : 특정 인덱스에 있는 요소를 제거
  - ✓ Insert() : 원하는 위치에 새요소를 삽입

```
ArrayList list = new ArrayList();  
list.Add(10);  
list.Add(20);  
list.Add(30);  
  
list.RemoveAt(1); // 20을 삭제  
list.Insert(1, 25); // 25를 1번 인덱스에 삽입, 즉, 10과 30 사이에 25를 삽입
```



# ArrayList 예제 코드

```
static void Main(string[] args)
{
    ArrayList list = new ArrayList();
    for (int i = 0; i < 5; i++)
        list.Add(i);

    foreach (object obj in list)
        Console.Write($"{obj} ");
    Console.WriteLine();

    list.RemoveAt(2);

    foreach (object obj in list)
        Console.Write($"{obj} ");
    Console.WriteLine();

    list.Insert(2, 2);
```

출력 결과:

0 1 2 3 4

0 1 3 4

0 1 2 3 4

0 1 2 3 4 abc def

```
foreach (object obj in list)
    Console.Write($"{obj} ");
Console.WriteLine();
```

```
list.Add("abc");
```

```
list.Add("def");
```

```
for (int i = 0; i < list.Count; i++)
    Console.Write($"{list[i]} ");
Console.WriteLine();
```

```
}
```





# ArrayList : 다양한 형식의 객체를 입력

- ArrayList 는 다양한 형식의 객체를 담을 수 있음
  - ✓ Add( ), Insert( ) 메소드는 object 형식의 매개변수 사용
  - ✓ 모든 데이터 형식을 object 형식의 매개변수로 입력 받는 것이 가능

```
public virtual int Add(object? value);  
public virtual void Insert(int index, object? value);
```

- object 데이터 형식은 값 형식의 데이터를 처리하기 위해 박싱, 언박싱 수행
  - ✓ 박싱, 언박싱은 적지 않은 오버헤드를 요구하는 작업
  - ✓ C# 은 오버헤드를 줄이기 위해 일반화 컬렉션 제공



# Queue

- Queue 는 대기열, 즉 기다리는(대기) 줄(열)이라는 뜻
- 입력은 오직 뒤에서, 출력은 앞에서만 이루어짐
- Queue 방식의 활용
  - ✓ OS 에서 CPU가 처리해야할 작업을 정리할 때
  - ✓ 프린터가 여러 문서를 처리할 때
- Queue 의 주요 메소드
  - ✓ Enqueue() : 가장 마지막에 있는 항목 뒤에 새 항목을 추가
  - ✓ Dequeue() : 제일 앞에 있던 항목이 출력

```
Queue que = new Queue();  
que.Enqueue(1);  
que.Enqueue(2);  
  
int a = (int)que.Dequeue();  
Console.WriteLine(a);    // 출력 값: 1
```



# Queue 예제 코드

```
static void Main(string[] args)
{
    Queue que = new Queue();
    que.Enqueue(1);
    que.Enqueue(2);
    que.Enqueue(3);
    que.Enqueue(4);
    que.Enqueue(5);

    while (que.Count > 0)
        Console.WriteLine(que.Dequeue());
}
```

출력 결과:

1  
2  
3  
4  
5



# Stack

- Queue 와 반대로 먼저 들어온 데이터가 나중에 출력 (First In – Last Out)
- 나중에 들어온 데이터는 먼저 출력 (Last In – First Out)
- Stack 의 주요 메소드
  - ✓ Push() : 데이터를 넣을 때, 데이터를 위에 쌓음
  - ✓ Pop() : 제일 위에 쌓여 있는 데이터를 꺼냄
- Pop() 을 호출하여 데이터를 꺼내면, 그 데이터는 컬렉션에서 제거
  - ✓ 그 아래 있던 데이터가 가장 위로 올라옴

```
Stack stack = new Stack();  
stack.Push(1); // 최상위 데이터는 1  
stack.Push(2); // 최상위 데이터는 2  
stack.Push(3); // 최상위 데이터는 3  
  
int a = (int)stack.Pop(); // 최상위 데이터는 다시 2  
Console.WriteLine(a); // 출력값: 3
```



# Stack 예제 코드

```
static void Main(string[] args)
{
    Stack stack = new Stack();
    stack.Push(1); // 최상위 데이터는 1
    stack.Push(2); // 최상위 데이터는 2
    stack.Push(3); // 최상위 데이터는 3
    stack.Push(4); // 최상위 데이터는 4
    stack.Push(5); // 최상위 데이터는 5

    while (stack.Count > 0)
        Console.WriteLine(stack.Pop());
}
```

출력 결과:

5  
4  
3  
2  
1



# Hashtable

- 키(key)와 값(value)의 쌍으로 이루어진 데이터를 다룰 때 사용
  - ✓ 영어사전을 예로 들어, "book"을 키로, "책"을 값으로 입력
  - ✓ 탐색속도가 빠르고 사용하기 편리함
- 해싱 (Hashing)
  - ✓ 원하는 데이터를 찾을 때, 키를 이용해 단번에 데이터가 저장된 컬렉션 내 주소 계산
  - ✓ 어떤 형식이든 키로 사용할 수 있음 (예: int, float, string)
  - ✓ 배열에서 인덱스를 이용해 배열요소에 접근하는 것에 준하는 탐색속도를 가짐

```
Hashtable ht = new Hashtable();  
ht["book"] = "책";  
ht["cook"] = "요리사";  
ht["tweet"] = "지저귀다";  
  
Console.WriteLine(ht["book"]); // 출력값: 책  
Console.WriteLine(ht["cook"]); // 출력값: 요리사  
Console.WriteLine(ht["tweet"]); // 출력값: 지저귀다
```





# Hashtable 예제 코드

```
static void Main(string[] args)
{
    Hashtable ht = new Hashtable();

    ht["하나"] = "one";
    ht["둘"] = "two";
    ht["셋"] = "three";
    ht["넷"] = "four";
    ht["다섯"] = "five";

    Console.WriteLine(ht["하나"]);
    Console.WriteLine(ht["둘"]);
    Console.WriteLine(ht["셋"]);
    Console.WriteLine(ht["넷"]);
    Console.WriteLine(ht["다섯"]);
}
```

출력 결과:

one  
two  
three  
four  
five



# 컬렉션 초기화

- ArrayList, Queue, Stack은 배열을 이용해 초기화 가능
  - ✓ 컬렉션 생성자를 호출할 때 배열의 객체를 매개변수로 입력
  - ✓ 컬렉션 객체는 해당 배열을 바탕으로 내부 데이터를 채움

```
int[] arr = { 123, 456, 789 };  
ArrayList list = new ArrayList(arr);    // 123, 456, 789  
Stack stack = new Stack(arr);           // 789, 456, 123  
Queue queue = new Queue(arr);           // 123, 456, 789
```

- ArrayList 는 컬렉션 초기자를 이용해 초기화 가능
  - ✓ 컬렉션 초기자는 생성자를 호출할 때, 생성자 뒤에 { 와 } 사이에 컬렉션 요소의 목록을 입력하여 사용

```
ArrayList list2 = new ArrayList() { 11, 22, 33 };
```



# Hashtable 예제 코드

```
static void Main(string[] args)
{

    int[] arr = { 123, 456, 789 };

    ArrayList list = new ArrayList(arr);
    foreach (object item in list)
        Console.WriteLine($"ArrayList: {item}");
    Console.WriteLine();

    Stack stack = new Stack(arr);
    foreach (object item in stack)
        Console.WriteLine($"Stack: {item}");
    Console.WriteLine();
```

```
Queue queue = new Queue(arr);
foreach (object item in queue)
    Console.WriteLine($"Queue: {item}");
Console.WriteLine();
```

// 컬렉션 초기화 사용

```
ArrayList list2 = new ArrayList() { 11, 22, 33 };
foreach (object item in list2)
    Console.WriteLine($"ArrayList2: {item}");
}
```

출력 결과:

ArrayList: 123  
ArrayList: 456  
ArrayList: 789

Stack: 789  
Stack: 456  
Stack: 123

Queue: 123  
Queue: 456  
Queue: 789

ArrayList2: 11  
ArrayList2: 22  
ArrayList2: 33



# Hashtable 초기화

- Hashtable 초기화를 위해 딕셔너리 초기자(Dictionary Initializer) 이용

```
Hashtable ht = new Hashtable()  
{  
    [ "하나" ] = 1,  
    [ "둘" ] = 2,  
    [ "셋" ] = 3  
};
```

- Hashtable 초기화를 위해 컬렉션 초기자도 이용 가능

```
Hashtable ht2 = new Hashtable()  
{  
    { "하나", 1 },  
    { "둘", 2 },  
    { "셋", 3 }  
};
```





**Thank you!**